File I/O

```
#include <fstream>
```

WRITE

```
{
    ofstream of("FILENAME");   // opens
    if (!of) {                 // of.operator!()
        std::cerr << "could not open for writing" << endl;
    } else {
        of << "...";
    } // Destructor closes the file
}
```

READ

```
{   ifstream ifs("FILENAME");
    if (!if) {
        // ... Error out
    } else {
        std::string s;
        while (true) {
            getline(ifs, s); // Reading into s
            if (ifs.eof()) {
                break;
            } else {
                // ...
            }
        }
    }
} // Destructor closes the file.
```

Function Template : represents a family of functions.

- specifies a pattern from which actual functions can be generated by substituting

template arguments for template parameters.

⇒ a template is NOT a function

```
template < typename T >
T max ( const T & x, const T & y)
{
    if (x < y)
        return y;
    else
        return x;
}
```

Not a reserved keyword

Template parameter

→ Must be defined in a Header file.
(if you want it to be used by multiple source files)

```
int main() {
    max <int> (3, 4);
    max <float> (5.6, 6.8);
    ...
}
```

template argument

Multiple template parameters:

```
template < typename T1, typename T2 >
```

Non-type template parameter

```
template <int N>
void foo ()
{
    int array[N];
    :
}
```

Compile time known

```
template < typename T, T value >
```

```
T max (const T & x)
{
    return (x < Value ? Value : x);
}
```

Class Template: a construct that represents a family of classes.

— specifies a pattern from which actual classes can be generated.

⇒ Parameterized classes.

```
template <typename T>
class A /* <T> */
{
  --private:
      T mVal;
};

template <typename T>
A<T>::A (const T& aValue) {
  ---
}
```

A<int> a;